リアルタイム画像計測のための多数粒子情報の並列抽出アーキテクチャの設計と実装

Design and Implementation of Parallel Extraction Architecture for Information of Numerous Particles in Real-time Image Measurement

○ 渡辺 義浩（東大） 小室 孝（東大） 鏡 慎吾（東大） 石川 正俊（東大）
Yoshihiro WATANABE, Takashi KOMURO, Shingo KAGAMI and Masatoshi ISHIKAWA
University of Tokyo, 7–3–1, Hongo, Bunkyo-ku, Tokyo 113–8656, Japan

画像内の多数粒子の情報を高速に抽出可能なアーキテクチャを提案する. ここでの粒子情報とは, 画像モーメントによって取得される種々の特徴量を指している. 提案するアーキテクチャは, 1 度に複数粒子のモーメントを並列に抽出する構成をとる. これにより, 多数の粒子が画像内にある場合の演算量を大幅に削減している. 加えて, 非同期処理によって, 処理の高速化を図るものである. 本アーキテクチャによって, 高いフレームレートのリアルタイム処理においても, より多くの情報を取得することが可能となる. この点は, 画像計測を主とした, 応用分野において効果が大きく, 様々なアプリケーションへの展開が期待できる. 本発表では, アーキテクチャ の構成と FPGA へ実装した結果を示す.
Key Words: Image Measurement, Moment Extraction, Particle Information, Real-time Image Processing

1. はじめに

画像計測は, 非接触での計測や柔軟性の高い操作を利点として持っており, 様々な応用展開を可能とする重要な技術である. 特に, 本研究では, ロボティクス, マルチメディア, 検査応用などの視覚情報のフィードバックを伴うアプリケーションへの展開に主眼を置く. このような応用においては, 処理のリアルタイム化が求められる. すなわち, 計測データを 1 フレーム時間内に獲得し, 状態変化に敏速に対応することが必須である.

同時に, 環境変化への敏速な対応や高速な現象の観測を実現するためには, 視覚装置のフレームレートの向上が必要である. この点は, 近年開発がなされている高速ビジョン [1, 2, 3] の導入によって可能となる. 高速ビジョンとは, 例えば NTSC 30Hz に代表されるビデオレートを大幅に上回る高いフレームレートでの撮像を可能とする視覚装置を指している.

高速ビジョンによる観測レートの向上とリアルタイムでのデータ獲得の 2 点は, 新たなアプリケーションを生み出す強力な要素になり得ると考えられる. しかし, 多くの視覚処理において, その演算量は膨大である. 従来のビデオレートにおいても, リアルタイム処理を保証することは困難であった. この点は応用展開にとって大きな障害である. これに対し, 本稿では, 高いフレームレートにおいてもデータのリアルタイム獲得を可能とする, 新しいアーキテクチャの確立を図る.

本稿では, 計測のための画像特徴量として, 粒子情報に着目する. ここでの粒子情報とは, 個々の対象領域の大きさ, 重心位置, 姿勢, 大まかな形状などを指している. これらの情報は, 各粒子の画像モーメントから取得できる. 提案するアーキテクチャは, 高いフレームレート下においても, 動画像より多数の粒子情報を抽出することを可能とするものである. このような処理の実現は, 広い応用分野において高い有用性が期待できる.

具体的なアプリケーションの 1 つとして, Particle Image Velocimetry (PIV) [4] のリアルタイム化が考えられる. PIV は, 多数粒子の状態を解析することで, 流れ場の情報を測定する画像計測である. 従来より, PIV はバッチ処理として実現されている. 本稿では, 特にモーメントから得られる粒子情報による解析を考える. PIV のリアルタイム化によって, 流体, 運動計測の精度や柔軟性の飛躍的な向上が期待できる. 例えば, ロボットビジョンへの応用が考えられる. 他にも, リアルタイムでモーメントより得られる種々の情報を利用し, 高速に移動する原生動物群の顕微鏡観測, 検査処理, 工業応用などのアプリケーションへの展開も想定している.

提案するアーキテクチャは, 並列処理によって, 1 度に複数領域のモーメントを同時にに抽出する構成をとる. この並列抽出により, 演算量を大幅に削減することが可能である. 加えて, 本アーキテクチャは, 非同期処理によって処理の高速化を図る. 本稿では, アーキテクチャの構成とその評価を述べるとともに, FPGA へ実装した結果を示す.

2. 設計方針

本稿での粒子情報とは, 画像モーメントによって取得される, 粒子の種々の特徴量を指している. 画像モーメントは, 画像解析やパターン認識において効果が非常に高く, 幅の広い応用分野において重要な役割を担っている [5]. 画像 $\boldsymbol{I}$ に対する $(i+j)$ 次の画像モーメント m_{ij} は, 式 (1) で表される.

$$m_{ij} = \sum_x \sum_y x^i y^j I(x,y) \tag{1}$$

ここで, $I(x,y)$ は画素 (x,y) における画像 $\boldsymbol{I}$ の値である. 画像モーメントより, 画像内での対象領域の大きさ, 重心位置, 姿勢, おおまかな形状などの様々な特徴量を取得することができる. しかし, 式 (1) に示されるように, モーメント抽出は, 画面全体の情報を利用する演算量の大きい処理である. 高速な抽出を行うためには適した処理系の導入が必要である. また, 本研究では, 多数の粒子に対する演算を目的としている. アーキテクチャは, 粒子情報の演算と多数対象への適用の 2 面を考慮した上での最適化が求められる.

モーメント抽出の多大な演算量に関しては, 並列処理の導入

によって解消される. これは, 画素単位の並列処理と総和を計算する回路の導入により, 計算時間の大幅な短縮が可能であるためである [6, 7]. 汎用ディジタルビジョンチップ [8] と呼ばれる視覚デバイスでは, この構成をとるモーメント抽出アーキテクチャが採用されており, 1kHz といった高いフレームレート下でも, 画像からのモーメント抽出を可能としている. これまでに, ビジョンチップによる高速な特徴量取得は, ロボット制御 [9, 10], 顕微鏡下の生物観察 [11, 12], 検査処理など様々なアプリケーションに適用されている.

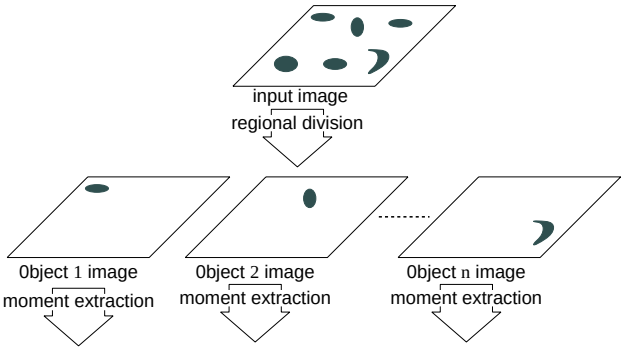

Fig. 2.1 Conventional Method of Separating Image with Multiple Objects

このように, 並列処理でのモーメント抽出は, 高速化の点で有効な手法であり, アプリケーションにおいてもその効果が示されている. 但し, これらのアプリケーションはいずれも単一の対象観測に基づいたものであった. これは, 従来アーキテクチャが, 対象領域を 1 つだけ含む画像への演算の適用を前提としていることに起因する. 従来アーキテクチャは, 画面全体のモーメント演算を基本としているため, 複数の領域が画像内に存在する場合には, 入力画像を 1 つずつの領域を含む画像へ分解する必要がある. その上で, Fig. 2.1 に示されるように, 逐次的にモーメント抽出を適用する必要がある. この点は, モーメント抽出のための演算量が対象数に比例して増大することを意味しており, 多数の粒子が存在するような画像を扱う場合には高速性の点で致命的な問題である.

一方で, 画素並列の構造を持った回路上では, 処理をパターンに応じて分離することで, 複数粒子の情報を同時に抽出することが可能であると考えられる. 本稿では, これを粒子情報の並列抽出と呼ぶ. この並列抽出によって, 演算量の大幅な削減を図る.

提案するアーキテクチャは, 従来手法によるモーメント演算と, 並列抽出の両者の統一化によって, 多数粒子情報の高速抽出を可能とするものである. 次節において, アーキテクチャの詳細を述べる.

3. 多数粒子情報の並列抽出アーキテクチャ

3-1 基本概念

まず, 並列機構によるモーメント抽出に関して概略を述べる. 並列プロセッサアレイ内では, 各列のモーメントが下位ビットから上位ビットへビットシリアルに演算される. アレイ外では, 並列に演算された各列の結果を受け取り, 対象画像のモーメントが出力される. また, この手法は, 任意の画素データを選択

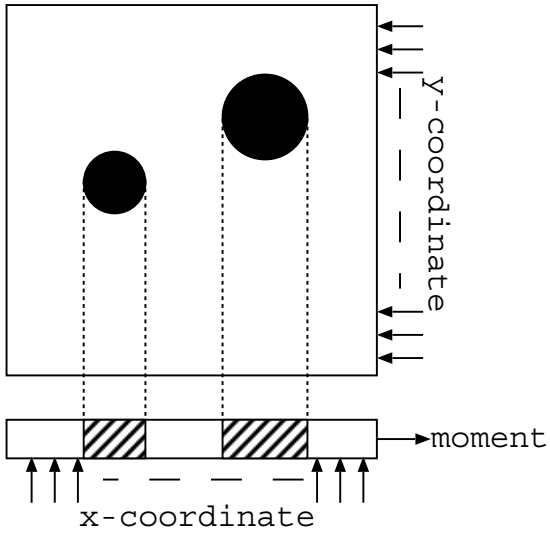

Fig. 3.1 Parallel Extraction

しながら, ビットシリアルに演算を行うことで, 高次のモーメントに関しても, 総和演算だけで計算することができる特徴を持つ.

並列抽出では, 並列処理の構造を利用し, 複数の対象の処理を同時に実行する. Fig. 3.1 に, 並列抽出の概念図を示す. 図においては, アレイ内の演算は列毎に並列に行われている. 図のように, 画像内で列方向に互いに重なっていない複数の対象は, 互いの演算を干渉しない. そのため, このような配置をとる対象に対しては, 同時に演算を実行できることがわかる.

アレイ内からの演算結果は, 対象間の空白を利用して分離することができる. また, アレイ外の演算は内部の演算に比べて十分に高速化できる. これは, アレイ内の演算はビットシリアルに行われるのに対し, 外部の演算は回路規模に関して制約が緩く, 任意の構成をとることができるためである. このため, 全体の計算時間は, 並列抽出によるアレイ内のものに依存すると言える.

このように, 並列抽出の導入によって, モーメント演算自体の高速化を保ったまま, 従来は対象数に依存していた演算量を大幅に削減することができると考えられる. 但し, 並列抽出は, 任意のパターンから演算方向に重なりを持たない対象群を選択する前処理を必要とする. この選択処理に関しては次節で述べる. また, 並列抽出による具体的な削減量に関しては 3-3 節においてシミュレーションを通して述べる.

3-2 並列抽出対象群の選択

本節では, 並列抽出対象群を選択するためのアルゴリズムを述べる. 選択処理では, 画面内における全対象の配置関係を知る必要がある. これは, 画面全体に渡る走査を伴う画像処理となり, 多大な計算時間を要する. しかし, 並列抽出による高速化の効果を保つためには, 選択処理がモーメント演算と同程度の時間で完了しなければならない. この点に関しては, 並列アルゴリズムによる処理の実行が有効である.

また, これと同時に非同期処理の利用によって高速化を図る. 本稿での選択処理は, 反復後の結果の収束が保証されている繰り返し演算によって構成される. このような繰り返し演算で記

述することにより, クロックによる同期を排除した形で処理を実行することが可能である. 画素間の通信は非同期に行われ, その速度は信号伝搬の平均遅延のみによって決定される. このため, 画像上の広い範囲に渡る処理も, 高速に実行可能であると考えられる.

まず, 選択処理における前提を述べる. 本処理は 2 値画像に対して適用されるものとする. また, 非零の上下左右の 4 近傍画素同士は, 同一対象領域内のものとする. 対象の形状に関して, 次のような定義を設ける. 本稿では全ての形状を, 演算方向に凹であるものとそうでないものの 2 つに分類する. 本処理では, 凹形状を凸化することで形状を統一する処理を備えている. これによって, 形状の違いが影響する処理の煩雑化を避ける. 但し, 本処理のために, 凹形状の対象とその陥没部分において重なっている領域同士は, 同一の対象として見なされる. 本処理は, これを領域分割の例外として持つものとする.

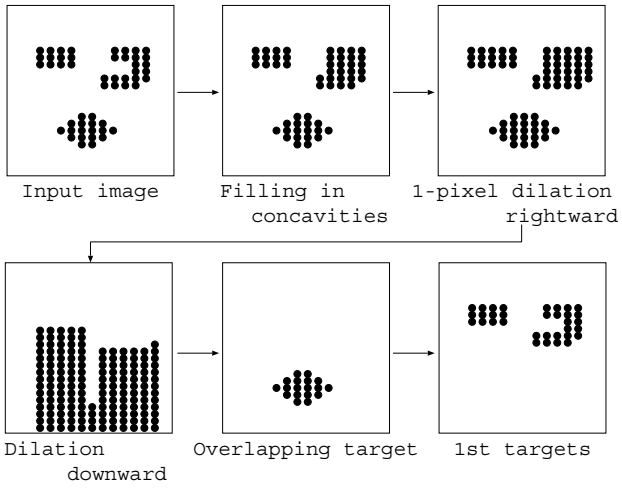

Fig. 3.2 Target Selection for Parallel Extraction

Fig. 3.2 に本処理の概要図を示す. また, 各画素において実行されるアルゴリズムの詳細を以下に示す. 式において, $\boldsymbol{f}$ は入力画像である. また, 添字 k, n を含む処理は, 繰り返し演算を意味している. 添字 k は単一の式での繰り返し処理を, n は複数の式に渡る繰り返し処理を示している. 添字のとれたものは, 繰り返し演算の結果の収束値である. さらに, x, y が負の場合, それを引数とする値は零であるとする.

Step 1. Filling in concavities

$$\begin{cases} g^0(x,y) = f(x,y) \\ g^{n+1}(x,y) = g^n(x,y) \cup i^n(x,y) \\ \quad \begin{cases} h^n_{k+1}(x,y) = \overline{g^n(x,y)} \cap \{g^n(x,y-1) \cup h^n_k(x,y-1)\} \\ \qquad\qquad \cap \{g^n(x-1,y) \cup g^n(x+1,y)\} \\ i^n_{k+1}(x,y) = h^n(x,y) \cap \{i^n_k(x,y+1) \cup g^n(x,y+1)\} \end{cases} \end{cases}$$

Step 2. 1-pixel dilation rightward

$$d(x,y) = g(x,y) \cup g(x-1,y)$$

Step 3. Dilation downward

$$e_{k+1}(x,y) = e_k(x,y-1) \cup \{d(x,y-1) \cap \overline{d(x,y)}\}$$

Step 4. Extraction of overlapping targets

$$\begin{cases} j_0(x,y) = e(x,y) \cap d(x,y) \\ j_1(x,y) = f(x,y) \cap \{j_0(x,y) \cup j_0(x+1,y) \\ \qquad\qquad \cup j_0(x,y-1) \cup j_0(x,y+1)\} \\ j_{k+1}(x,y) = f(x,y) \cap \{j_k(x,y) \cup j_k(x-1,y) \\ \qquad\qquad \cup j_k(x+1,y) \cup j_k(x,y-1) \cup j_k(x,y+1)\} \end{cases}$$

Step 5. Selection of targets for parallel extraction

$$k(x,y) = f(x,y) \cap \overline{j(x,y)}$$

Step 6. Parallel extraction

–Apply parallel extraction to an image $\boldsymbol{k}$.

Step 1 において, 凹形状の陥没部分を埋めている. Step 2 から 4 によって, 演算方向において重なりを持つ対象を検出している. Step 5 において, 並列抽出が適用される対象群が選択される.

3-3 並列抽出による演算量削減効果

本節では, シミュレーションを通して, 並列抽出による演算量削減の効果を示す. シミュレーションにおいては, 画像中に存在する全ての粒子情報を取得するために必要なモーメント抽出の回数を計数した. ここでは, 全ての次元のモーメント抽出を 1 セットとして数えており, 計数した回数は, 並列抽出する粒子群の選択回数に相当する. 画像サイズは, 256×256 pixel^2 である. 画像パターンは, 半径 6 pixel 以下の円の大きさと位置をランダムに設定することで生成した.

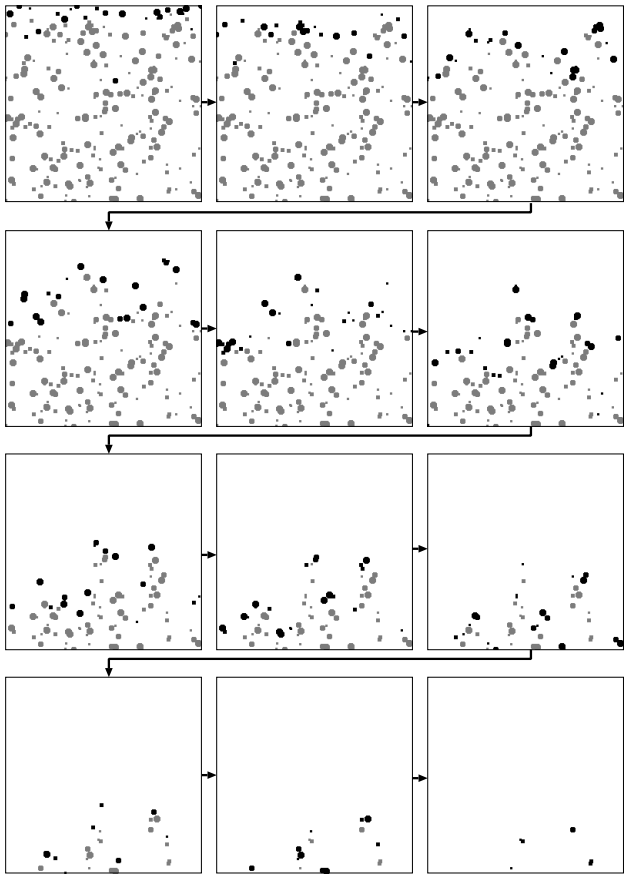

Fig. 3.3 Example of Parallel Extraction

Fig. 3.3 は, 画面内に 172 個の粒子を含む場合の並列抽出の様子を示している. 図において, 黒い粒子が並列抽出を適用される対象群である. 並列抽出の利用によって, 従来は 172 回の抽出処理が必要であったのに対して, 12 回のみの演算で全体の抽出処理が完了することがわかる.

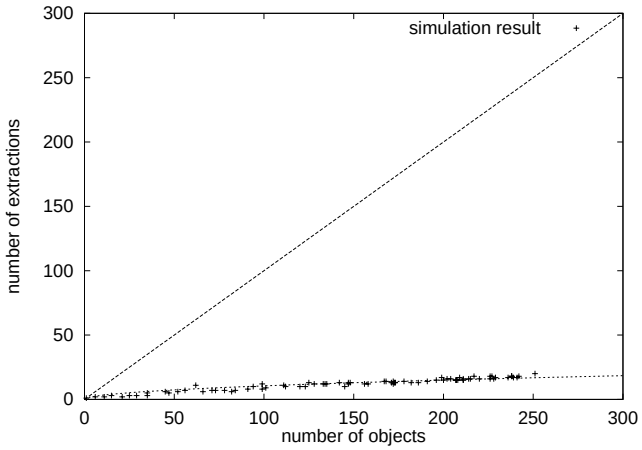

Fig. 3.4　Simulation Result

Fig. 3.4 は, 対象数と抽出回数の関係を表している. 点は, 並列抽出のシミュレーション結果をプロットしたものである. 曲線は, プロットされた点の近似曲線である. この曲線は, $(0,0),(1,1)$ を通ることを前提に, 指数関数による近似を用いて求めた. また, 直線は従来の場合を示している. 対象数を a, 抽出回数を b とすると, 曲線は $b = a^{0.511}$, 直線は $b = a$ である. 結果より, 並列抽出によって, 大幅な演算量の削減が可能であることが分かる.

3-4　アーキテクチャ評価

本節では, 提案するアーキテクチャの評価を行う. ここでは, 適用するアプリケーションの例として, PIV を想定する. 具体的な観測環境を次のように設定する. 観測粒子は, 速度 10 m/s で運動する. 観測領域は, 200×200 mm^2 とする. 4000 個程度の各粒子を径 10 pixel 程度で観察するために, 解像度は 1024×1024 pixel^2 とする. フレーム間の粒子移動を 5 pixel 以内に留めるために, フレームレートを 100 μs とする. 視覚装置に関しては, 解像度 352×288 pixel^2, フレームレート 10 kHz での撮像, 及び画像転送を可能とする研究事例 [13] が既にある. このため, ここで想定するリアルタイム PIV のための適当な視覚デバイスは, 十分に実現可能なものであると仮定する.

また, モーメント抽出の処理モデルを次のように設定する. アレイ内のモーメント抽出回路は, 処理要素をツリー接続することで構成されるものとする. 各要素は, 1 ビットの全加算器とキャリー保持のためのフリップフロップから成る. 画像は, $2^n \times 2^n$ pixel^2 (1024×1024 pixel^2 時は $n = 10$) の 2 値画像とする.

Fig. 3.1 に示されるように, x 座標値の乗算を伴うモーメントに関しては出力された各列の結果から得られる. このため, アレイ内で必要な演算は, y 座標値に関連するものだけでよい. 例えば, 0 次から 2 次のモーメント抽出を考えた場合, アレイ内で必要な演算は m_{00}, m_{01}, m_{02} となる. m_{10}, m_{20} に関しては m_{00} から, m_{11} に関しては m_{01} からそれぞれアレイ外にて演算が可能である. 1 クロック毎に 1 ビットずつシリアルに演算が行われるため, アレイ内の 0 次モーメント抽出には $(n+1)$ クロックサイクルを要する. 同様に, 1 次モーメントは $(2n-1)$ クロックサイクル, 2 次モーメントは $(3n-1)$ クロックサイクルを要する. したがって, 合計で $(6n-1)$ クロックサイクルを要する. クロック周期は, アレイ内のモーメントが抽出されるまでの時間として計算される. よって, 加算による遅延を 1 段あたり t_1 とすると, ツリー接続されているため, nt_1 となる. また, 3-1 節で述べたように, アレイ外の処理は, アレイ内に比べて十分に高速化できるものとする. 結果として, 0 次から 2 次のモーメント抽出のための全体の処理時間は, $nt_1(6n-1)$ となる.

次に, 選択処理に関して述べる. 選択処理の全体の処理時間を, 3-2 節で示した, Step 1, 3, 4 のそれぞれの繰り返し処理に要する時間の和で見積もる. 画素当たりの処理遅延は, t_2 とする. これは, それぞれの処理式における 1 度の反復のための実行時間に相当する. また, 全ての粒子と凸化処理において埋めるべき穴はそれぞれ, $A \times A$ pixel^2, $B \times B$ pixel^2 に収まるものとする. これより, 選択処理のための全体の処理時間は, $t_2(2B^2 + 2^n + 2A)$ と近似できる.

リアルタイム PIV への適用可能性を, 上記処理モデルより算出される処理時間によって考える. 3-3 節の結果に基づき, 粒子数 a に対する演算回数は, $\sqrt{a}$ とする. 各パラメータを $t_1 = t_2 = 1, A = 10, B = 3$ とする. 1024 × 1024 の並列度での実装が最も高速であるが, 他の実装形式も考えられる. 今回のアプリケーションの条件は, 入力画像全体をいくつかのブロックに分割し, 逐次それらのブロックを処理する形式でも満たすことができる. 今回の場合, 適切なブロックのサイズは, 256 × 256 pixel^2 である. $n = 8$ の場合, 0 次から 2 次までのモーメント抽出は 376 ns, 選択処理は 284 ns を要する. これら 2 つの処理は同時に行うことができるため, 全体の処理時間をいずれか長い方の時間を用いて見積もる. 分割されたそれぞれのブロックに, 等しく a 個の粒子が存在した場合, 全体の処理時間は $(16 \times 376 \times \sqrt{a})$ ns となる. 画像全体の総粒子数は, $16a$ である. 以上より, 100 μs 以内に, 4096 個の粒子に関して 2 次までのモーメントを得ることがことができることが分かる.

この結果は, 本アーキテクチャの実現によって, 高いフレームレートで動画像を扱うようなリアルタイム画像計測への発展が可能であることを十分に裏付けている. さらに, この処理モデルにおいては, 総和回路に関してパイプライン化などの余地があり, さらなる高速化が可能である.

4.　FPGA への実装

本節では, 提案するアーキテクチャを試作した結果を示す. 今回は, 動作検証を目的としており, ハードウエアとして, FPGA を選択した. 回路はハードウエア記述言語 Verilog を用いて記述した. 全体システムのブロック図を Fig. 4.1 に示す. 今回, アレイは 32 × 32 の構成をとる.

Fig. 4.2 に, アレイ内の各列のモーメント抽出回路のブロック図を示す. 3-4 節で述べたように, アレイ内の総和回路は, 1 ビット全加算器とキャリー保持のためのフリップフロップで構成される要素をツリー接続したものを用いた. それぞれの要素は, 画素のデータを受け取り, 列の総和を出力する. また, アレ

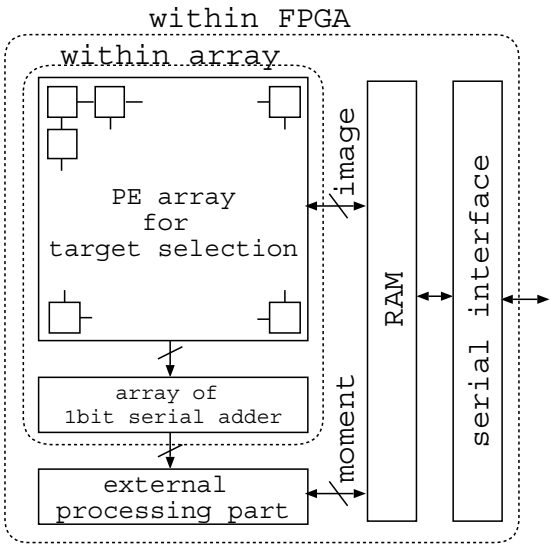

Fig. 4.1 Structure of Overall System

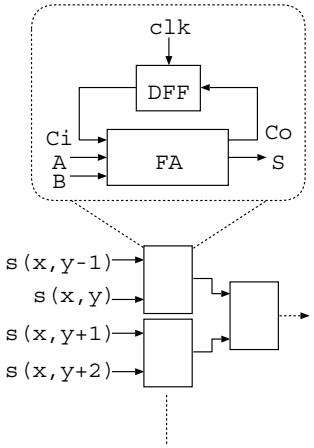

Fig. 4.2 Structure of Moment Extraction Part

イ外のモーメント抽出回路に関しては, アレイ内の演算に比べて, 高速に実行できる形である必要がある. しかし, 今回は簡単のため, 1 ビットシリアルに演算する構成をとった. 行方向に複数段用意することで, 複数の対象に関する演算結果を同時に取得できることができる. 今回の実装では, 4 段用意した. これらの回路は, アレイ内で演算された各列の総和を受け取り, 各対象の総和結果を出力する. 今回の回路は, 粒子情報として, 0 次, 及び 1 次モーメントを出力する.

Fig. 4.3 は, 選択処理のための各画素の回路である. 図に示されるように, 構成を切り替えることで複数の処理を実行できる形式をとった. これによって, 回路規模の縮小を図っている. 実装した回路において, 各画素の処理要素は, 近傍画素と非同期に通信を行い, 高速に処理を実行した.

ターゲットとした FPGA は, Xilinx 社の Virtex-II Pro シリーズの XC2VP50 である. XC2VP50 は, 53,136 個のロジックセルを内蔵している. また, 232 個のブロック RAM を内蔵しており, 総容量は 4,176kbit である. Fig. 4.4 に, 今回利用した FPGA ボードの写真を示す. 本ボードは, XC2VP50 を搭載した Virtex-II Pro FF1152 ボードである. 今回の実装では, スピードグレー

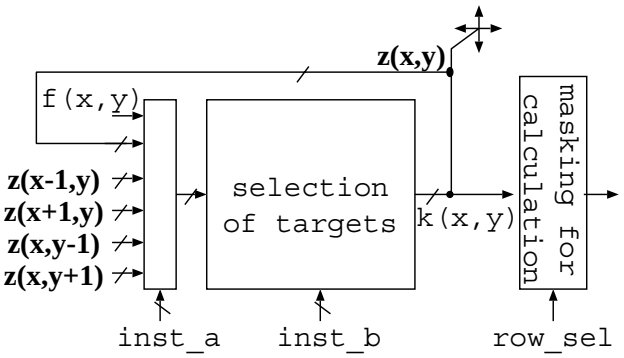

Fig. 4.3 Structure of Selection Element

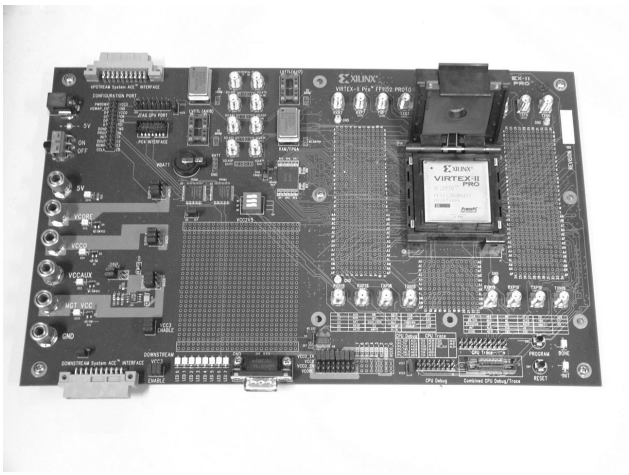

Fig. 4.4 FPGA Board for Implementation

ドは-7 とした. FPGA と計算機の間での, 画像の入力や処理結果の取得はシリアルポートを介して行った. 結果として, 32×32 の画素アレイ, 及びアレイ外の処理部の実装に, 上記 FPGA の約 38%のリソース (slice) を要した. 動作時に供給するクロック周波数は, 100MHz とした. このクロックは, 総和演算における制御に用いられている. ここでは, モーメント抽出の 1 段当たりの遅延を 2 ns としている. この値は, 3-4 節の t_1 に相当する.

任意のパターンの画像を与えた結果, 適切に並列抽出の対象を選択し, 正しいモーメントを出力することを確認した. リソースに関しては, 選択処理の回路構成に改善の余地があり, さらに集積度を高めることが十分に可能である. また, クロック周波数に関しても, 構成の最適化によって, 上記の遅延を 1 ns まで向上できる余地を残しており, 200MHz 相当まで高速化することができる.

以上より, FPGA での試作を通して, ハードウエアの最適化に関しては, まだ多くの余地が残されているが, 提案するアーキテクチャが想定される能力を満たし得るレベルで実現可能であることを示した.

5. おわりに

多数粒子情報の並列抽出アーキテクチャを提案し, その設計をした. また, FPGA へ実装し, その動作を確認した. 本アーキテクチャは, 高いフレームレートのリアルタイム処理が必要な局面においても, 十分な量の情報を画像より取得することを可

能とする. このような処理は, 特に画像計測を主として, 様々なアプリケーションにおいて有効性が高い.

今後は, 本アーキテクチャに基づく処理系をカメラと接続し, 動画像を扱えるシステムとして環境を拡張する. このリアルタイムシステムにより, PIV に基づいた流体, 運動計測, 多数の原生動物群の顕微鏡下観測, ロボット制御のための画像認識, 検査処理, 工業応用などの分野において, 従来不可能であったアプリケーションの実現を新たに目指していく予定である.

参考文献

1) http://www.micron.com/products/imaging/

2) http://www.dalsa.com

3) http://www.fillfactory.com/

4) M. Raffel 他: “PIV の基礎と応用 –粒子画像流速測定法, ” Springer 社, 2000.

5) M. K. Hu: “Visual Pattern Recognition by Moment Invariants,” IRA Transactions on Information Theory, Vol. IT-8, pp. 179-187, 1962.

6) 小室 孝, 石井 抱, 中坊 嘉宏, 石川 正俊: “デジタルビジョンチップのためのモーメント抽出アーキテクチャ, ” 電子情報通信学会技術報告, Vol.PRMU99-51, pp.17-22, 1999.

7) 石井 抱, 小室 孝, 石川 正俊: “ディジタルビジョンチップのためのモーメント計算法, ” 電子情報通信学会論文誌 D-Ⅱ, Vol.J83-D-Ⅱ, No.8, pp.1733-1740, 2000.

8) T. Komuro, S. Kagami and M. Ishikawa: “A New Architecture of Programmable Digital Vision Chip,” 2002 Symposium on VLSI circuits Proceedings, pp.266-269, 2002.

9) Y. Imai, A. Namiki, K. Hashimoto, and M. Ishikawa: “Dynamic Active Catching Using a High-speed Multifingered Hand and a High-speed Vision System,” Proceedings of 2004 IEEE International Conference on Robotics and Automation, pp.1849-1854, 2004.

10) T. Senoo, A. Namiki, and M. Ishikawa: “High-Speed Batting Using a Multi-Jointed Manipulator,” Proceedings of 2004 IEEE International Conference on Robotics and Automation, pp.1191-1196, 2004.

11) 奥 寛雅, 石井 抱, 石川 正俊: “マイクロビジュアルフィードバックシステム, ” 電子情報通信学会論文誌 D-Ⅱ, Vol.J84-D-II, No.6, pp.994-1002, 2001.

12) N. Ogawa, H. Oku, K. Hashimoto,and M. Ishikawa: “Motile Cell Galvanotaxis Control using High-Speed Tracking System,” Proceedings of 2004 IEEE International Conference on Robotics and Automation, pp. 1646-1651, 2004.

13) S. Kleinfelder, S. Lim, X. Liu and A. E. Gamal: “A 10000 Frames/s CMOS Digital Pixel Sensor,” IEEE Journal of Solid-State Circuits, Vol.36, No.12, pp.2049-2058, 2001.